Quadrilateral Mesh Generation using Hierarchical Templates

Antonio Carlos de Oliveira Miranda[1] and Luiz Fernando Martha[2]

[1] Department of Civil and Environmental Engineering, University of Braslia, SG-12 Building, Darcy Ribeiro Campus, DF, 70.910-900, Brazil
acmiranda@unb.br

[2] Department of Civil Engineering, Pontifical Catholic University of Rio de Janeiro, Rua Marqus de So Vicente 225, Gvea, Rio de Janeiro, RJ, 22453-900, Brazil lfm@tecgraf.puc-rio.br

Abstract: This paper describes a quadrilateral mesh generation algorithm ideally suited for transition subdomain meshes in the context of any domain decomposition meshing strategy. The algorithm is based on an automatic hierarchical region decomposition in which, in the last level, it is possible to generate quadrilateral elements with a conventional mapping strategy. In two dimensions, a subdomain is usually a triangle or a rectangle. In this algorithm, a subdomain with two boundary curves may also be allowed. Templates impose restrictions on the number of boundary curve segments of a subdomain to be meshed. The proposed hierarchical template scheme eliminates these restrictions, requiring only an even number of boundary segments. Other algorithms in the literature present similar characteristics. However, the implementation of the hierarchical decomposition and its templates presented here is quite simple compared to other approaches. Six high-level templates are considered for a subdomain, depending on the number of boundary curves and the number of segments on each curve. Several examples demonstrate that this simple idea may result in structured meshes of surprisingly good quality. We also show the possibility of obtaining different meshes for a subdomain with fixed boundary discretization by changing the corners between curves.

Keywords: template-based mesh; structured quadrilateral mesh; domain decomposition; mapping, transition mesh

1 Introduction

This paper describes a novel hierarchical template-based meshing scheme for generating good-quality quadrilateral meshes. This approach is ideally suited for transition subdomain meshes in the context of structured 2D or surface meshing strategies, such as mapping, submapping, sweeping, medial axis, auto-decomposition or user-assisted decomposition. One of the main drawbacks of these meshing schemes is the constraint on the number of subdomain boundary curve segments. For quadrilateral subdomains, the number of segments on opposite boundary curves must be equal, and, for triangular subdomains, the three boundary curves must have equal number of segments. In this environment, it is difficult to implement local mesh refinement without using non-structured hybrid subdomain meshes because any change in the number of segments of a boundary curve forces the propagation of this modification to opposite subdomain curves. The proposed hierarchical template-based meshing scheme produces quad-mapping transition meshes without any constraint on the number of boundary segments. The only requirement is that the total number of segments must be even, which is a general rule for any quadrilateral mesh [?, ?].

In the context of quadrilateral mesh generation, template is a pattern that describes how a single polygon can be decomposed into quadrilaterals. In two dimensions or in surface meshing, a single polygon is usually a triangle or a rectangle. Many finite element meshing algorithms use templates to some degree. For example, mapping techniques may be considered as simple templates. Initially, classical structured mapping strategies [?] were proposed, defining generalized curvilinear coordinate systems for closed, bounded and simply connected domains on the plane or in 3D surfaces. A similar approach was proposed in another work [?] for generated hexahedral meshes using body-oriented coordinates defined by three-dimensional regions bounded by six surfaces. In both works, transfinite mapping techniques were established for curvilinear coordinate systems in arbitrary domains to approximate complex surfaces and volumes. Haber et al. [?] and Haber and Abel [?] used transfinite mappings based on discrete boundary curves, and applied these techniques to two-dimensional and three-dimensional surface preprocessing programs. This discrete form of the mapping allows representing boundary geometries generically. The basic idea of these works is to use triangular and quadrilateral template meshes in a parametric space and map them to Cartesian space. Similar ideas were developed to generate three-dimensional meshes [?, ?]. It is interesting to observe that, in 1982, Cook and Oakes [?] presented examples of quadrilateral mesh grading algorithms for gradual or rapid element density transitions, but the algorithms were not formalized. Similar techniques were cited by Thompson [?].

Another meshing technique that employs templates is recursive domain subdivision using quadtree [?]. Yerry and Shephard [?] pioneered this technique, proposing templates to generate triangular and quadrilateral elements. Other works have been published following similar ideas with some improvements and modifications [?, ?, ?, ?]. In general, this mesh generation process is implemented in three stages. Initially, the domain's interior is filled with a quadtree that is recursively and locally refined according to given boundary refinement information. Care is taken to avoid adjacent quadtree cells that have a difference of more than one in tree depth. Then, templates that depend on cell adjacency are employed to mesh the interior cells. In a final stage, the region between the interior mesh and the boundary is also meshed using several meshing schemes, which might employ other types of templates. A recent work [?] describes a template scheme for meshing all quadrilateral elements with guaranteed quality while preserving features of the boundary. Analogous procedures are used for three-dimensional mesh generation using an octree [?, ?, ?, ?]. Schneiders et al. [?] presented original templates to generate hexahedral elements in octree cells, which were improved by Ito et al. [?].

There are many other meshing algorithms that use templates. Schneiders [?] reviewed the state of the art in quadrilateral and hexahedral mesh generation in 2000 and described many techniques that employ templates. Templates are naturally used in association with a domain decomposition strategy, in which a domain is decomposed into subdomains where a specific template is chosen to generate quadrilateral elements. For example, Nowottny [?] used a geometry-based optimization for selecting appropriate cuts dividing the domain and presented a set of meshing templates for triangular and rectangular polygons. In a sense, the hierarchical meshing scheme proposed in the present paper is a generalization of the templates presented by Nowottny. Mller-Hannemann [?] decomposed a coarse mesh of polygons in three-dimensional space into quadrilaterals. In these subdomains, mesh is generated with templates that satisfy prescribed local density constraints. Four quadrilateral templates are presented that are similar to some present here. Lizir et al. [?] proposed a template-based approach for generating quad-only meshes from 2D digital images. The same authors used the same technique to generate quad meshes from triangle surfaces [?]. In both works, they fill the subdomains with triangular and quadrilateral templates. In addition, many other approaches for 3D domain decomposition generate meshes using templates [?, ?, ?]. A commercial software for finite element analysis also employs templates for subdomain mesh transition using quadrilateral and hexahedral elements [?].

The mesh generation algorithm proposed in this work is also devised in the context of a domain decomposition meshing strategy. As mentioned, in two dimensions, a subdomain is usually a triangle or a rectangle. In this work, a subdomain with two boundary curves may be allowed. Templates impose restrictions on the number of boundary curve segments of a subdomain to be meshed. The proposed hierarchical template scheme eliminates these restrictions, requiring only an even number of boundary segments. The algorithm introduced by Müller-Hannemann [?] presents the same characteristic. However, our algorithm has a simpler and more direct approach than that algorithm.

Six high-level templates are considered here for a subdomain, depending on the number of boundary curves and the number of segments on each curve: three templates have four curves, two have three curves, and one has two curves. A boundary curve is given by a set of segment points (boundary nodes) and may include a group of geometric curves. A transition subdomain may have four, three, or even two boundary curves. Based on the input boundary data, the hierarchical scheme selects the target high-level template (classification) and recursively decomposes the subdomain into regions in which only quadrilateral templates may be adopted. The recursive decomposition results in subregions that are meshed using the classical quad-mapping scheme. The hierarchical recursive depth is three at most.

Although template-based quadrilateral mesh generation has already been studied by other authors, as described above, this work presents some contributions, namely:

- an automatic recursive region decomposition in which, in the last level, it is possible to generate quadrilateral elements with a conventional mapping strategy;
- the proposed template with three curves does not impose constraints on the number of subdivisions, such as the ones required by the tri-mapping technique [**?**], for instance;
- a new alternative template for subdomains with three curves for a particular case of curve subdivision;
- a new template for subdomains with two curves.

One of the main advantages of the proposed scheme is that it generates topologically equivalent meshes for subdomains with the same number of curves and boundary segments. This characteristic may be explored in volume sweeping meshing, since the source and target surface meshes are topologically equivalent. Another advantage is the possibility of obtaining different meshes for a subdomain with fixed boundary discretization by changing the corners between curves, as shown in the examples section. Finally, the implementation of the hierarchical decomposition presented here, and its templates, is quite simple when compared to other approaches.

2 Main Concept

Following the ideas of Haber et al. [**?**, **?**], the input data for the proposed quadrilateral mesh generation scheme on a subdomain is a discrete representation of boundary curves (polylines). As mentioned, this discrete form allows representing boundary geometries generically. This form of representation is quite simple, and may be implemented in any programming language as a vector of real numbers which is a sequential list of boundary points (or nodes) and the number of segments (or edges) in each boundary curve: $(x_1, y_1, z_1, x_2, y_2, z_2, ..., x_n, y_n, z_n)$. As also mentioned, to generate quadrilateral elements the total number of edges on the boundary must be even. A subdomain may be composed of four, three, or two boundary curves that do not intersect themselves. The number of boundary curves is indicated by

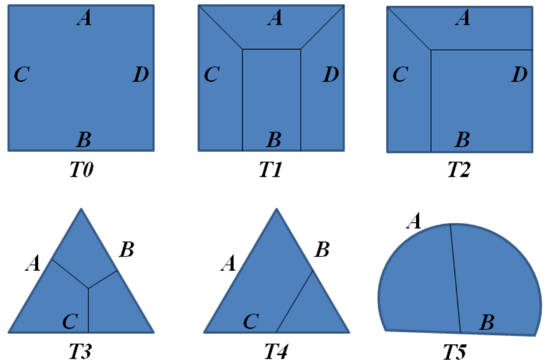

Fig. 1. Templates used to decompose regions and their nomenclature.

the number of corner nodes, which are given by a set of indices to the input boundary coordinate vector.

Figure **??** shows the set of templates considered in this work, which are used to decompose a region in subregions. They consist of two templates with four curves ($T1$ and $T2$), two templates with three curves ($T3$ and $T4$), and one template with two curves ($T5$). The letters A, B, C, and D in Figure **??** correspond to the number of edges in each boundary curve. Note that template $T0$ does not decompose the region; it is used only to generate quadrilateral elements through the conventional mapping method in which $A = B$ and $C = D$.

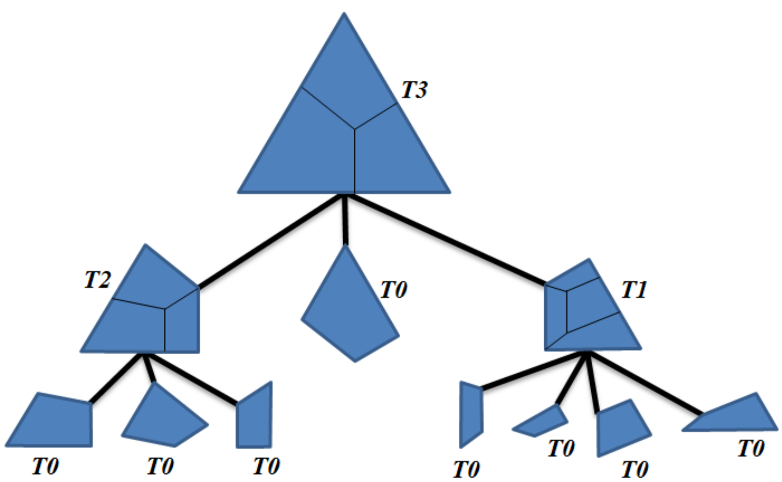

Fig. 2. Example of hierarchical decomposition of templates to generate quadrilateral elements.

The prior selection (first level) of one of the templates in Figure **??** depends on number of edges on each curve. If the number of edges on opposite sides is equal, then template $T0$ is selected, and quadrilateral elements are generated by the conventional mapping method. If it is not possible to use template $T0$ in the first level, one of the five other templates is selected. Each of these templates decomposes the first level subdomain into regions (second level), and a new template is selected for each region. This process is repeated recursively for each region until a subregion can be meshed using template $T0$. Due to this recursive process, the proposed template-based quadrilateral mesh generation can be understood as a *hierarchical decomposition*. The whole scheme was devised in such a way that the hierarchical recursive depth, i.e. the number of levels, is three at most. For example, Figure **??** shows a subdomain composed by three boundary curves. In the first level, template $T3$ is selected. In the second level, three different templates are selected for each subregion ($T2$, $T0$, and $T1$). In the third and last level, template $T0$ is selected for all subregions, which are the leaves in the hierarchical decomposition.

A key point in the hierarchical decomposition meshing scheme is the selection of a template to be used in a region. This selection is explained by means of a pseudo-code, which is shown in Figure **??**. Given the number of curves in a region and their number of edges (A, B, C, and D), this code returns the selected template. The algorithm described in Figure **??** is straightforward. First, the selection is based on the number of boundary curves. Then, the selection is based on the number of edges of each curve. The result will be a non-valid selection if the total number of subdivision edges is not even (a null value is returned). If the selection results in template $T0$, the corresponding region is meshed (conventional mapping) and the recursive process for that region stops. If the selected template is other than $T0$, the region is divided and the pseudo-code is used to select the template for each resulting subregion. The process is repeated recursively until $T0$ is selected for all subregions. To obtain the final subdomain mesh, the meshes of all $T0$ subregion leaves are merged.

3 Implementation Details

The equations presented by Gordon & Hall [**?**] for transfinite mapping of surface patches with four and three curves are used here to compute the position of any interior point generated by the hierarchical decomposition scheme. Considering that the input to the algorithm is a set of edges on the boundary curves, as described in the previous section, the discrete transfinite mapping presented by Haber et al. [**?**, **?**] is conveniently applied in this context. The mapping expressions are reproduced in equation (1) for the bilinear projector and in equation (2) for the trilinear projector. Therefore, in a more general form, the proposed scheme can be applied to 3D surface patches. In this case, a generated internal point is projected to the closest point on the surface.

```
Function getTemplate (numCurves, A, B, C, D)
 If numCurves = 2 Then
    If ((A + B) mod 2) = 0 Then
      Return Template T5
    End If
 Else If numCurves = 3 Then
    If ((A + B + C) mod 2) = 0 Then
      If shape is similar to a sliver triangle Then
        Return Template T4
      Else
        Return Template T3
      End If
    End If
 Else If numCurves = 4 Then
    If A = C And B = D Then
      Return Template T0
    Else If A = C And ((B + D) mod 2) = 0 Then
      Return Template T1
    Else If B = D And ((A + C) mod 2) = 0 Then
      Return Template T1
    If ((A + C) mod 2) = ((B + D) mod 2) Then
     Return Template T2
    End If
 End If
 Return NULL
End Function
```

Fig. 3. Pseudo-code to select a template based on the number of curves and their subdivision.

$$\begin{aligned} F(u,v) = &(1-v)\psi_1(u) + v\psi_2(u) + (1-u)\xi_1(v) + u\xi_2(v) \\ &-(1-u)(1-v)F(0,0) - (1-u)vF(0,1) \\ &-uvF(1,1) - u(1-v)F(1,0) \end{aligned} \tag{1}$$

$$\begin{aligned} T(u,v,w) = \frac{1}{2}\Bigg[&\left(\frac{u}{1-v}\right)\xi(v) + \left(\frac{w}{1-v}\right)\eta(1-v) + \left(\frac{v}{1-w}\right)\eta(w) \\ &+ \left(\frac{u}{1-w}\right)\psi(1-w) + \left(\frac{w}{1-u}\right)\psi(u) \end{aligned}$$

$$+\left(\frac{v}{1-u}\right)\xi(1-u) - w\psi(0) - u\xi(0) - v\eta(0)\Bigg] \qquad (2)$$

A key aspect of the proposed hierarchical decomposition process of a region is defining the number of edges that will be used on the boundaries of each subregion. The number of edges is defined based on the lengths of the boundary curves. These lengths are computed in 3D using the given discrete polyline geometric information on each curve. The following paragraphs detail, for each adopted template, the decomposition process and the computation of the number of edges on the boundaries of each resulting subregion.

Template $T1$, shown in Figure **??**(a), is applied when the number of edges of a pair of opposite curves is equal ($C = D$), and the number of edges of the other pair of opposite curves is different ($A \neq B$). As required, the values of A and B must satisfy the restriction $[(A+B)mod2] = 0$. Considering $B > A$, the number of edges b is given by $b = (A-B)/2$. Values of u_1, u_2, v_1, v_2 in parametric space, see Figure **??**(a1), are computed as:

$$u_1 = \frac{d_1}{d_3}, u_2 = \frac{d_2}{d_3}, v_1 = \frac{d_1}{d_1 + d_4}, v_2 = \frac{d_2}{d_2 + d_5} \qquad (3)$$

in which d_1, d_2, d_3, d_4, and d_5 are lengths in Cartesian space, as shown in Figure **??**(a2). The parametric values given by equation (3) result in quadrilateral elements of better shape quality generated in each subregion. All subregions generated by template T1 have the final template $T0$, with the following distribution of boundary edges:

- Subregion 1, $A \times b$ edges;
- Subregion 2, $b \times C$ edges;
- Subregion 3, $A \times C$ edges;
- Subregion 4, $b \times C$ edges.

Template $T2$, shown in Figure **??**(b1), is used when the number of edges of opposite curves is not equal, that is, $A \neq B$ and $C \neq D$. However, the evenness property requires the number of edges to be $[(A+B+C+D)\,mod2] = 0$. In Figure **??**(b1), $B > A$ and $D > C$, and the number of edges b is obtained from the expression $b = Min(A-B, C-D)/2$. Values of u_1 and v_1 in parametric space, see Figure **??**(b1), are computed by:

$$u_1 = \frac{d_1}{d_3}, v_1 = \frac{d_2}{d_4} \qquad (4)$$

in which d_1, d_2, d_3 and d_4 are lengths in 3D space, as shown in Figure **??**(b2). The subregions generated by template $T1$ have the following distribution of boundary edges:

- Subregion 1, $A \times b$ edges, with final template $T0$;
- Subregion 2, $b \times C$ edges, with final template $T0$;

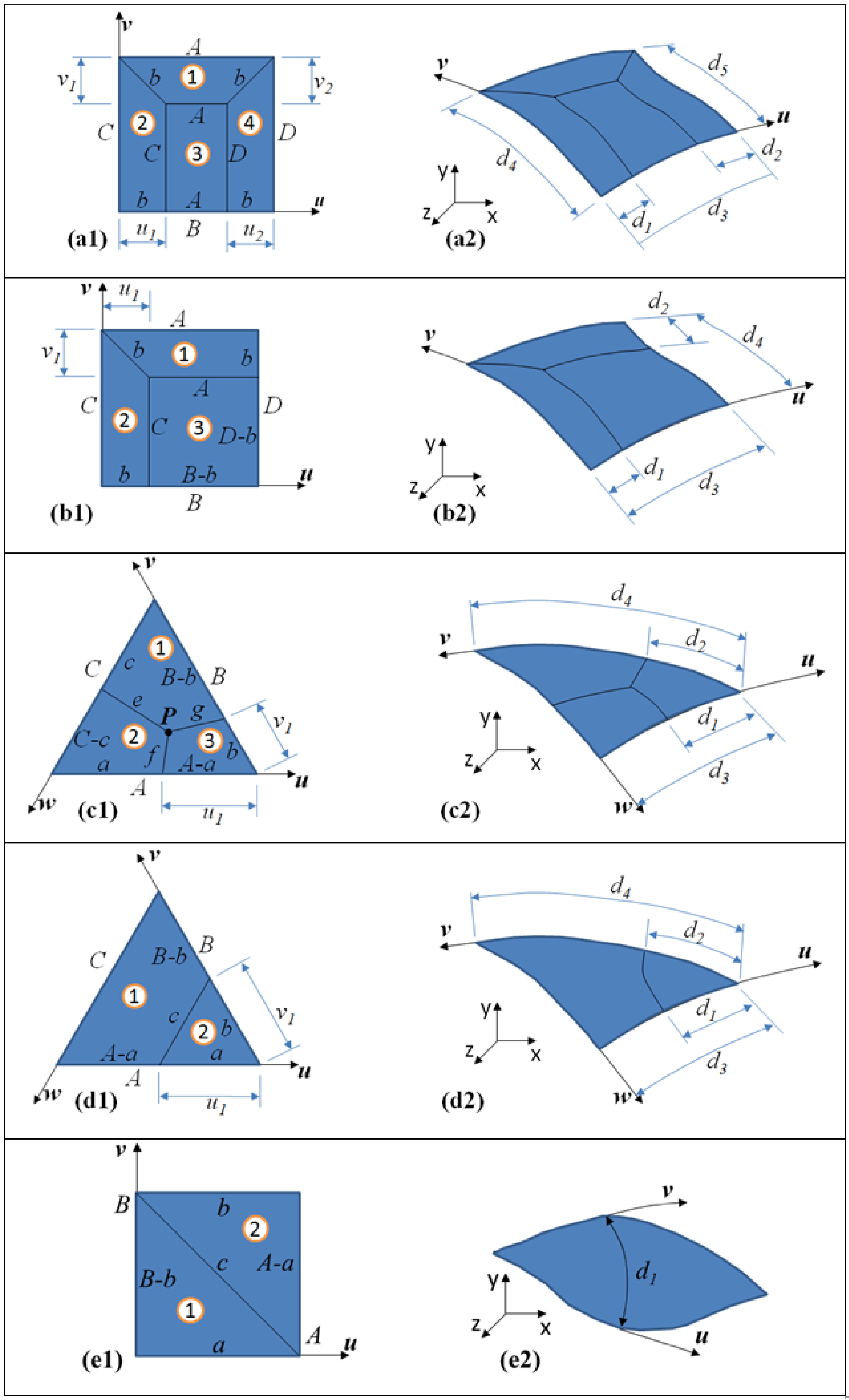

Fig. 4. Decomposition of the proposed templates in parametric and in Cartesian spaces.

- Subregion 3 with two possibilities: (1) If $A = (B - b)$ and $C = (D - b)$, the final template is $T0$; (2) If $A \neq (B - b)$ or $C \neq (D - b)$, the final template is $T1$, which is decomposed recursively.

Both templates $T1$ and $T2$ have been presented by other authors [**?**, **?**, **?**, **?**]. However, the templates presented here are more flexible because they may be applied recursively. This is one of the main advantages of the proposed meshing scheme, which turns out to be a natural way to apply templates. This may be noticed by comparing the proposed templates with the ones of a commercial software [**?**], for example, which imposes further restrictions on curve subdivision.

Template $T3$, as shown in Figure **??**(c1), consists in decomposing the region into three subregions of four curves. The procedure used here is similar to that used in trimapping [**?**]. However, here it is extended to use templates in a hierarchical manner. The problem with trimapping is that it presents the following restriction:

$$A + B > C + 2, B + C > A + 2, C + A > B + 2. \tag{5}$$

The numbers of edges a, b, and c, as shown in Figure **??**(c1), are achieved as follows:

$$a = (A + B - C)/2, b = (B + C - A)/2, c = (C + A - B)/2. \tag{6}$$

```
Input A, B, C
Output a, b, c, e, f, g
Comment: "compute # of edges on curves"
a = (A + B - C) / 2
b = (B + C - A) / 2
c = (C + A - B) / 2
minEdge = Min (a, b, c)
Comment: "adjust a, b, and c if necessary"
If minEdge ≤ 0 Then
  offset = 1 - minEdge;
  If a ≤ 0 Then a = a + offset Else a = a - offset
  If b ≤ 0 Then b = b + offset Else b = b - offset
  If c ≤ 0 Then c = c + offset Else c = c - offset
End If
Comment: "compute internal # of edges"
If a > (B - b) Then e = a Else e = B - b
If b > (C - c) Then f = b Else f = C - c
If c > (A - a) Then g = c Else g = A - a
```

Fig. 5. Pseudo-code to obtain the number of edges in template $T3$.

The procedure proposed here to template $T3$ does not necessarily conform to equation (5). When this restriction is not satisfied, an offset correction is applied, resulting in the number of edges given by equation (6). Figure **??** shows a pseudo-code to this procedure. Given the number of edges in each boundary curve (A, B, and C) of the original region, the number of edges in each internal curve (a, b, c, d, e, f, and g) is obtained. Initially, the values of a, b, and c are calculated using equation (6). If the restrictions in equation (5) are not obeyed, one of these calculated values will be equal to or smaller than zero, and all values need to be adjusted by an offset correction. The offset is a unit subtracted from the lower calculated value (a, b, or c). Then, the values of a, b, and c are adjusted with the following rule: if a value is smaller than or equal to zero, add the offset to this value; otherwise, subtract the offset from this value. The numbers of edges e, f, and g are obtained from the largest number of edges in the adjacent opposite curves, as shown in Figure **??**(c1) and in the pseudo-code of Figure **??**. For a simple example with $A = B = 2$ and $C = 10$, the first values obtained are $a = -3$, $b = c = 5$, resulting in an offset equal to 4; and, subsequently, resulting in $a = b = c = e = g = 1$ and $f = 9$.

Template $T4$ is proposed here to be used as an alternative to template $T3$ when the number of edges of one curve is much smaller than the number of edges of the other two curves. In this situation, template $T3$ may not provide good results in some cases. In Figure **??**(d1), assume that $C < A$ and $C < B$. One criterion that can be used for selecting $T4$ instead of $T3$ is $kC \leq A$ and $kC \leq B$, where k may be an integer at least greater than 2, $k \geq 2$. This value should be chosen according to the needs of each application. The values of a, b, and c, in Figure **??**(d1), initially can be set to C (the smallest number of edges among the input boundary curves). Note that the second subregion must satisfy the restriction $[(a+b+c)mod2] = 0$. When this restriction is not satisfied, the values of a and b must be adjusted. This is done using a very simple procedure: if $A > B$, $a = a + 1$; otherwise, $b = b + a$. Values of u_1 and v_1 in parametric space are computed similarly to template $T3$, where d_1, d_2, d_3, and d_4 are lengths obtained in Cartesian space, as shown in Figure **??**(d2). The subregions generated by template $T4$ are:

- Subregion 1 with two possibilities: (1) If $(A - a) = (B - b)$, the final template is $T0$; (2) If $(A - a) \neq (B - b)$, the final template is T1.
- Subregion 2, with $a \times b \times c$ edges, to use template $T3$.

Template $T5$, shown in Figure **??**(e), is used when the domain has only two curves. A domain with two curves could be considered as a domain with three curves, dividing the curve with the largest number of edges into two curves [**?**]. However, the proposed template $T5$ divides each of the two boundary curves into two further curves to form a bilinear mapping in parametric space, as shown in Figure **??**(e1). Then, the region is divided into two subregions with three curves. The number of internal edges c can be calculated or reported by the application. A possible calculation is to take the distance d_1, shown in Figure ??(e2) in Cartesian space, and divide it by the average size of all boundary edges. Restrictions and must be satisfied. The subregions generated by template $T5$ are:

- Subregion 1, with $(B - b) \times a \times c$ edges, to use template $T3$;
- Subregion 2, with $(A - a) \times b \times c$ edges, to use template $T3$.

4 Examples

This section presents some examples of the application of the proposed quadrilateral mesh generation scheme in regions of simple shapes. The main objectives of these examples are: (1) to show the behavior of the hierarchical decomposition when the number of edges on curves is modified; (2) to illustrate the impact of selecting different boundary curves for a region with fixed boundary subdivision; and (3) to show that meshes generated in different regions with the same number of curves and subdivisions are topologically equivalent.

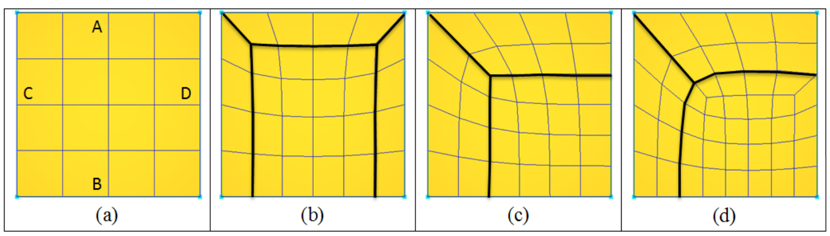

Fig. 6. Set of meshes for a square domain; example 1.

In the first two examples, shown in Figures ?? (example 1) and ?? (example 2), the impact of varying the number of edges of the boundary curves on template selection and on the final mesh is studied. In the images, thicker lines represent the boundaries of the resulting subregions. Table ?? shows the input numbers of edges, the root template, and the branch templates used in the examples illustrated by these figures. Figure ?? presents a set of meshes for a square region, in which templates $T1$ and $T2$ are used. The most complex situation is found in Figure ??(d), in which all input boundary curves have different subdivisions. In this case, a branch $T1$ template is used in the bottom-right subregion.

Example 2 is shown in Figure 9: an equilateral triangle. In this example, the meshes shown in Figures ??(a), 9(b), and 9(h) are obtained in a similar way as the trimapping technique [?]. However, the meshes in Figures 9(c), 9(d), and 9(e) may only be generated using the proposed approach. The meshes of Figures ??(f) and 9(g) could be generated by template $T3$ similarly to

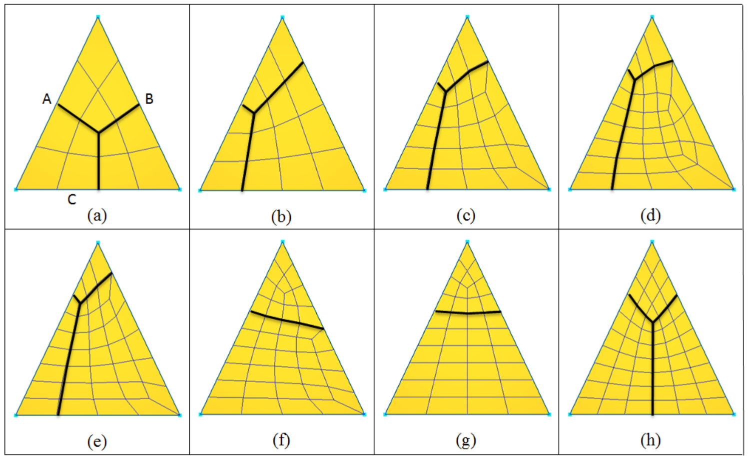

Fig. 7. Set of meshes for an equilateral triangular domain; example 2.

Table 1. Curve refinement and templates used in examples 1 and 2.

Figure	A	B	C	D	Root template	Branch templates
8(a)	4	4	4	4	$T0$	–
8(b)	4	6	4	4	$T1$	–
8(c)	4	6	4	6	$T2$	–
8(d)	4	8	4	6	$T2$	$T1$ (bottom-right region)
9(a)	4	4	4	–	$T3$	–
9(b)	6	4	4	–	$T3$	–
9(c)	8	4	4	–	$T3$	$T1$ (bottom-right region)
9(d)	10	4	4	–	$T3$	$T1$ (bottom-right region)
9(e)	10	6	4	–	$T3$	$T1$ (bottom-right region)
9(f)	10	8	4	–	$T4$	$T3$ (top region) and $T1$ (bottom region)
9(g)	10	10	4	–	$T4$	$T3$ (top region)
9(h)	10	10	6	–	$T3$	–

trimapping. However, template $T4$ is used here because it generates better results.

Examples 3 and 4, illustrated in Figures **??** and **??**, show a set of meshes generated for domains with a fixed number of edges on the boundary, but with different sets of boundary curves. The corners between boundary curves are defined by the round marks shown in these figures, and the decomposition of the root template in subregions is represented by the thicker lines. The number of curves, the root template, and the branch templates are listed in Table **??**. In order to assess the quality of the generated meshes, Table **??**

Table 2. Numbers of curves and templates used in examples 3 and 4.

Figure	# Curves	Root template	Branch templates
10(a)	4	$T2$	$T1$ (top-right region)
10(b)	3	$T3$	$T1$ (top-left region)
10(c)	3	$T4$	$T3$ (top-right region) and T1 (bottom region)
10(d)	3	$T3$	–
10(e)	2	$T5$	$T3$ (top-left and bottom-right regions)
10(f)	2	$T5$	$T3$ (top-left and bottom-right regions)
11(a)	3	$T4$	$T3$ (top region) and $T1$ (bottom region)
11(b)	3	$T3$	$T1$ (bottom-right region)
11(c)	2	$T5$	$2 \times T4 \Rightarrow 2 \times T3$ (top and bottom regions) and $2 \times T1$ (middle region)
11(d)	2	$T5$	$2 \times T3 \Rightarrow 2 \times T3$ (top and bottom regions) and $2 \times T1$ (middle-right region)

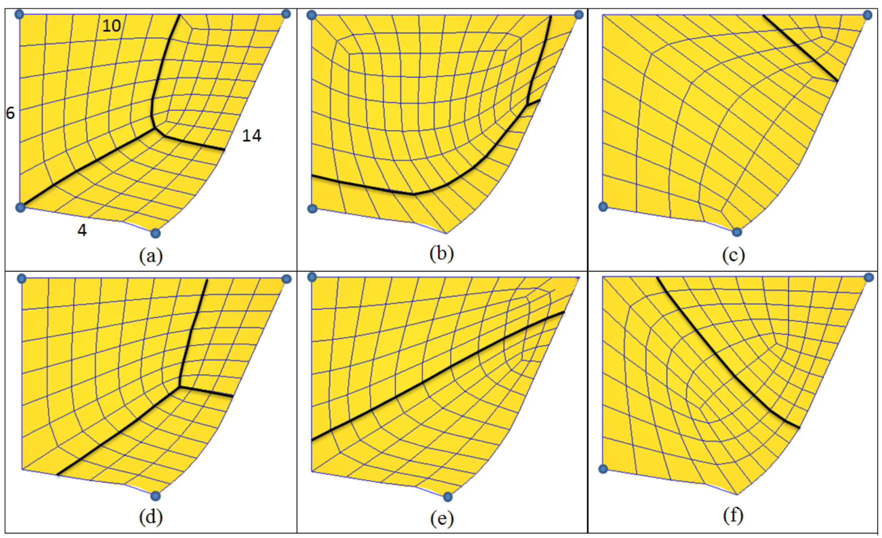

Fig. 8. Set of meshes when different boundary curves are specified for a domain, example 3.

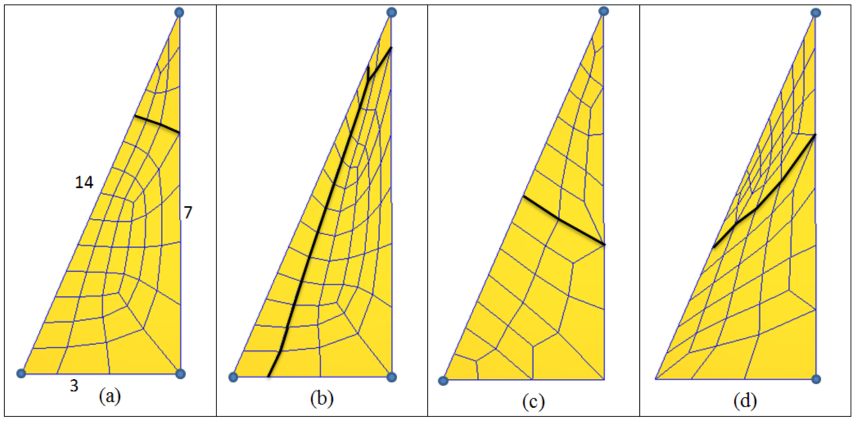

Fig. 9. Set of meshes when different boundary curves are specified for a domain; example 4.

Table 3. Mesh metrics for examples 3 and 4.

Figure	# of nodes	# of elements	$\alpha_{average}$	α_{max}	α_{min}	Standard Deviation
10(a)	116	98	0.495	0.894	0.127	0.187
10(b)	119	101	0.402	0.849	0.042	0.223
10(c)	88	70	0.267	0.832	0.017	0.243
10(d)	109	91	0.637	0.955	0.101	0.208
10(e)	120	102	0.420	0.911	0.032	0.232
10(f)	130	112	0.380	0.870	0.056	0.240
11(a)	63	50	0.284	0.764	0.024	0.187
11(b)	69	56	0.256	0.860	0.002	0.231
11(c)	45	32	0.306	0.728	0.020	0.225
11(d)	68	55	0.160	0.707	0.006	0.163

lists shape quality metrics for each mesh. The distortion metric used in this table is the one proposed by Lee and Lo [**?**];

$$\beta = \frac{\alpha_3 \times \alpha_4}{\alpha_1 \times \alpha_2}, \tag{7}$$

in which α_i is adopted as the internal angle computed for each of the four resulting triangles in the i_{th} corner, sorted in descending order of magnitude, $\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \alpha_4$. The shape quality metric has a valid interval between 1.0 and 0.0, with high quality elements, those close to a right rectangle, having values close to 1.0.

The analysis of mesh results illustrated in Figures **??** and **??** are based on the metric values summarized in Table 3. The natural number of boundary curves of the domain shown in Figure **??** is four. Considering this information, the mesh generated by the proposed procedure is shown in Figure **??**(a). Alternatively, this domain can be meshed considering three curves, as shown in Figures **??**(b, c, d) or two curves, as in Figures **??**(e, f). Surprisingly, the best result regarding element shape quality is a situation with three boundary curves, as shown in Figure **??**(d). A similar behavior is observed for the domain with three curves shown in Figure **??**. Although the domain has clearly three boundary curves, the best result is obtained when two boundary curves are considered, as in Figure **??**(c). Note also that the use of template $T4$, Figure **??**(a), presents better shape quality results than the use of template $T3$, thereby demonstrating the need for an alternative template for three curve domains. In both examples, the boundary is naturally defined with four or three curves. However, better results are obtained with an alternative number of curves. Defining the best choice of boundary curves for a given domain is outside the scope of this work.

5 Conclusion

This work presented a hierarchical template-based quadrilateral meshing scheme. Six templates were presented: three templates with four curves, two with three curves and one with two curves. The main concept of the proposed approach is to decompose a region into subregions, in a recursive and hierarchical way, until achieving a subregion in which it is possible to generate quadrilateral elements using the bilinear mapping technique. Meshes of all subregions are merged to obtain one final mesh.

Template decomposition is based on discrete bilinear and trilinear projectors of parametric coordinates on boundary curves. Details of the main procedures were described to help readers implement them. Some of the contributions are:

- Two existing templates from the literature for a region with four curves were improved in terms of domain decomposition;
- An existing template (trimapping technique) for a region with three curves was extended, and a restriction was removed by adding an offset correction;
- An alternative template with three curves was proposed;
- A new template with two curves was proposed.

Some examples were presented, showing the behaviour of the proposed meshing scheme when the number of edges of boundary curves is modified, and when different curves are considered as input for a region. These examples demonstrate how useful the proposed alternative template is for a region with three boundary curves. In addition, they demonstrate that the original number of curves of certain domains does not necessarily result in the best mesh when applying the proposed approach. Other possibilities must be tested. Finally, the proposed templates were used to create topologically equivalent source and target surface meshes in volumetric mesh generation using a sweeping technique.

Acknowledgements

The authors would like to thank the National Council for Scientific and Technological Development (CNPq), University of Braslia, the Computer Graphics Technology Group (Tecgraf) and Pontifical Catholic University of Rio de Janeiro (PUC-Rio) for the financial support and for providing the necessary space and resources used during the development of this work.

References

1. Cook W.A. and Oakes W. R. (1982) Mapping Methods for Generation Three-Dimensional Meshes. Computer In Mechanical Engineering 67–72
2. Mitchell S.A. (2000) High Fidelity Interval Assignment. International Journal of Computatinal Geometry and Applications 10(4):399–415
3. Gordon W. J. and Hall C. A. (1973) Contruction of Curvilinear Co-ordinate Systems and Aplications to Mesh Generatio. International Journal for Numerical Methods in Engineering 7:461–477
4. Cook W. A. (1974) Body Oriented (natural) Co-ordinates for Generating Three Dimensional Meshes. International Journal for Numerical Methods in Engineering 8:27–43.
5. Haber R., Shephard M.S., Abel J.F., Gallagher R.H., and Greenberg D.P. (1981) A General Two-Dimensional, Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mapping. International Journal for Numerical Methods in Engineering 17:1015–1044
6. Haber R. and Abel J.F. (1982) Discrete Transfinite Mappings for Description and Meshing of Three-Dimensional Surfaces Using Interactive Computer Graphics. International Jornal For Numerical Methods in Enginnering 18:41–66
7. Perucchio R., Ingraffea A.R., Abel and J.F. (1982) Interative Computer Graphics Preprocessing for Three-Dimensional Finite Element Analysis. International Journal for Numerical Methods in Engineering 18:909–926
8. Thompson J.F., Soni B.K., and Weatherill N.P. (1999) Handbook of grid generation, CRC Press
9. Samet H. (1984) The Quadtree and Related Hierarchial Data Structures. ACM Computer Surveys 6(2):187–260
10. Yerry M. and Shephard M. (1983) A Modified Quadtree Approach To Finite Element Mesh Generation. IEEE Computer Graphics and Applications 3(1):39–46
11. Baehmann P. L., Wittchen S. L., Shephard M. S., Grice K. R., and Yerry M.A. (1987) Robust, geometrically based, automatic two-dimensional mesh generation. International Journal for Numerical Methods in Engineering 24(6): 1043–1078
12. Yiu K.F.C., Greaves D.M., Cruz S., Saalehi A., and Borthwick A.G.L. (1996) Quadtree grid generation: Information handling, boundary fitting and CFD applications. Computers & Fluids 25(8):759–769

13. Smith, R. J., and Johnston L. J. (1996) Automatic grid generation and flow solution for complex geometries. Reston, VA, ETATS-UNIS: American Institute of Aeronautics and Astronautics.
14. Liang X., Ebeida M.S., and Zhang Y. (2010) Guaranteed-quality all-quadrilateral mesh generation with feature preservation. Computer Methods in Applied Mechanics and Engineering 199:2072–2083
15. Yerry M.A. and Shephard M.S. (1984) Automatic three-dimensional mesh generation by the modified-octree technique. International Journal for Numerical Methods in Engineering 20(11):1965–1990
16. Zhang H. and Zhao G. (2007) Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method. Finite Elements in Analysis and Design, 43(9):691-704
17. Ito Y., Shih A.M., and Soni B.K. (2009) Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. International Jornal For Numerical Methods in Enginnering 77:1809-1833
18. Schneiders R., Schindler R., and Weiler F. (1996) Octree-based generation of hexahedral element meshes. In: Proceedings of 5th International Meshing Roundtable 1:205–215
19. Schneiders R. (2000) Algorithms for Quadrilateral and Hexahedral Mesh Generation. In: Proceedings of the VKI Lecture Series on Computational Fluid Dynamic, VKI-LS 2000-4
20. Nowottny D. (1997) Quadrilateral Mesh Generation via Geometrically Optimized Domain Decomposition. In: Proceedings of 6th International Meshing Roundtable, (1):309–320
21. Mller-Hannemann M. (2000) High Quality Quadrilateral Surface Meshing Without Template Restrictions: A New Approach Based on Network Flow Techniques. International Journal of Computational Geometry and Applications, 10(3):285–307
22. Lizir M., Siqueira M., Daniels J., Silva C., and Nonato L. (2011) Template-based quadrilateral mesh generation from imaging data. The Visual Computer 27(10):887–903
23. Daniels J., Lizier M., Siqueira M., Silva C.T., and Nonato L.G. (2011) Template-based quadrilateral meshing. Computers & Graphics 35(3):471–482
24. Staten M.L., Kerr R.A., Owen S.J., Blacker T.D., Stupazzini M., and Shimada K. (2010) Unconstrained plasteringHexahedral mesh generation via advancing-front geometry decomposition. International Journal for Numerical Methods in Engineering 81(2):135–171.
25. Mitchell S.A. (1999) The All-Hex Geode-Template for Conforming a Diced Tetrahedral Mesh to any Diced Hexahedral Mesh. Engineering with Computers, 15:228–235
26. Yamakawa S. and Shimada K. (2001) Hexhoop: Modular Templates For Converting A Hex-Dominant Mesh To An All-Hex Mesh. In: Proceedings of 10th International Meshing Roundtable (1):.235–246
27. Ansys (2012) Documentation for Ansys - Transition Mapped Quadrilateral Meshing. Available from: http://www.kxcad.net/ansys/ANSYS/ansyshelp/Hlp_G_MOD7_4.html.
28. Lee C. K. and Lo S. H. Lo (1994) A new scheme for the generation of a graded quadrilateral mesh. Computers and Structures (52):847–857
29. Scott M.A., Earp M.N., and Benzley S.E. (2010) Adaptive sweeping techniques. Engineering with Computers (26):317–325